09-28-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventorship..... Vetrivelkumaran et al.
Applicant..... Microsoft Corporation
Attorney's Docket No. MS1-658US
Title: Systems and Methods for Controlling the Number of Clients that Access a Server

TRANSMITTAL LETTER AND CERTIFICATE OF MAILING

To: Commissioner of Patents and Trademarks,
Washington, D.C. 20231

From: James R. Banowsky (Tel. 509-324-9256; Fax 509-323-8979)
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Specification–title page, plus 39 pages, including 51 claims and Abstract
2. Transmittal letter including Certificate of Express Mailing
3. 4 Sheets Formal Drawings (Figs. 1-4)
4. Return Post Card

Large Entity Status [x] Small Entity Status [ ]

Date: 9/26/00 By: James R. Banowsky
Reg. No. 37,773

CERTIFICATE OF MAILING

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Postal Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed below, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable) EL685271515

Date: 9/26/2000 By: Dana L. Calhoun
Dana L. Calhoun

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Systems and Methods For Controlling The Number Of Clients That Access A Server

Inventors:

Vellore T. Vetrivelkumaran
Raju Gulabani
Steve Falcon
Neel Malik

ATTORNEY'S DOCKET NO. MS1-658US

TECHNICAL FIELD

The systems and methods described herein relate to server-client network systems. More particularly, the described implementations relate to controlling the number of unique clients that can access server software in a server-client network.

BACKGROUND

Developers of operating systems and software applications require that users of these systems have a license to use the system. The license indicates that the user has paid for the right to use the software. Business enterprises that license software for use in enterprise systems usually enter into a concurrent access license with the software developer. A concurrent users license allows a limited number of unique users to use the software. Such licenses, however, can be virtually impossible to enforce.

Some network software systems are configured to allow only a certain number of users to access the software at the same time. For example, an operating system may be configured to allow only ten users to access the operating system at any given time. However, limiting access to ten users at the same time does not necessarily limit access to the operating system to ten unique users. Typical use of a system will see many users logging on and off a system for various periods throughout a day. One user may log onto the system for ten minutes and log off. Another user may log on for an hour before logging off. Therefore, it is possible that many more than ten users may access a system even though no more than ten users are logged onto the system at any given time.

Furthermore, software developers sometimes develop smaller versions of large software packages. These smaller versions - which are less expensive than the larger versions - are targeted for small businesses that do not need the capabilities of the larger version or that may not be able to afford the larger systems. If the software developer cannot limit the number of unique users that access the system, then a larger enterprise may use a version that was designed for a smaller enterprise and avoid using a more costly software package. Such misuse by an enterprise harms the software developer.

SUMMARY

Methods and systems are described herein that limit the number of clients that can access server software to a pre-defined limit. A server operating system - or other software system running on a server - includes a communications protocol filter that monitors transmissions between the server and multiple clients connected to the server via a network.

When the communications filter detects a packet header that identifies a client attempting to connect to the server, the communications filter identifies a network address associated with the client from the packet and attempts to locate the network address in a table that contains a network address for each client that has previously accessed the server. If the network address for the client is in the table, then the server processes the communications packets received from the client.

If the network address for the client is not listed in the table, then the server references a client limit field in server memory. The client limit field contains a configurable client limit value that denotes the number of clients that are allowed

to access the server. The client limit value is encrypted to prevent illegal alteration of the client limit. If the number of entries in the table is less than the value in the client limit field (i.e., the number of clients having access to the server is less than the allowable limit), then the client is allowed to access the server and the network address for the client is added to the table.

In some systems, a network address that is assigned to a client may expire after a certain period, such as when the client has been inactive for a certain period of time. When the client becomes active again, a new network address is assigned to the client. Often, the new network address will be identical to the original network address, but there are cases when the network addresses are different.

To accommodate these types of systems, access to the server is not simply denied if the number of client network addresses in the table meets or exceeds the client limit. If the number of network addresses in the table is greater than or equal to the client limit when a client attempts to access the server, and the client's network address is not in the table, the communications filter determines if the client requesting server access is a previous client that has a new network address. In one implementation, the communications filter transmits a signal to each network address that is listed in the table. If a client at a network address does not respond to the signal, the communications filter assumes that the network address that does not provide a response is an old network address of the client attempting to access the system. Therefore, it removes the old network address from the table, inserts the network address of the client requesting access into the table, and allows the client to access the server. If, however, all network addresses respond to the signal, then the client limit has been reached and allowing access to the requesting client would exceed the limit. The client is, therefore, denied access to

the server and a 'limit exceeded' event occurs. Upon occurrence of this event, a signal is sent to indicate that the client limit has been exceeded and the packet is not processed.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of exemplary methods and arrangements of the present invention may be had by reference to the following detailed description when taken in conjunction with the accompanying drawings wherein:

Fig. 1 is an exemplary computer system on which the present invention may be implemented.

Fig. 2 is an illustration of a server having a communications filter, the server communicating with several clients over a network.

Fig. 3 is a block diagram of a server-client network system implemented in accordance with the described embodiments.

Fig. 4 is a flow diagram outlining a method for controlling the number of clients having access to a server in a server-client network system.

DETAILED DESCRIPTION

The invention is illustrated in the drawings as being implemented in a suitable computing environment. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, to be executed by a computing device, such as a personal computer or a hand-held computer or electronic device. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Exemplary Computer Environment

The various components and functionality described herein are implemented with a number of individual computers. Fig. 1 shows components of typical example of such a computer, referred by to reference numeral 100. The components shown in Fig. 1 are only examples, and are not intended to suggest any limitation as to the scope of the functionality of the invention; the invention is not necessarily dependent on the features shown in Fig. 1.

Generally, various different general purpose or special purpose computing system configurations can be used. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

The functionality of the computers is embodied in many cases by computer-executable instructions, such as program modules, that are executed by the computers. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Tasks might also be performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media.

The instructions and/or program modules are stored at different times in the various computer-readable media that are either part of the computer or that can be read by the computer. Programs are typically distributed, for example, on floppy disks, CD-ROMs, DVD, or some form of communication media such as a modulated signal. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable media when such media contain instructions programs, and/or modules for implementing the steps

described below in conjunction with a microprocessor or other data processors. The invention also includes the computer itself when programmed according to the methods and techniques described below.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

With reference to Fig. 1, the components of computer 100 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISAA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as the Mezzanine bus.

Computer 100 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by computer 100 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. "Computer storage media" includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of

information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more if its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 100, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Fig. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

The computer 100 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through an non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface such as interface 150.

The drives and their associated computer storage media discussed above and illustrated in Fig. 1 provide storage of computer-readable instructions, data structures, program modules, and other data for computer 100. In Fig. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and

information into the computer 100 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball, or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

The computer may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 100, although only a memory storage device 181 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, the computer 100 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 100 typically includes a modem

172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 100, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Fig. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Fig. 2 is a simplified illustration of a computer network system 200 that implements one or more of the described implementations. The computer network system 200 includes a server 202 and several client computers that communicate with the server 202 over a network 204. For purposes of the present discussion, the network 204 is a local area network (LAN), although other types of networks may be used. Client A 206, Client B 208, Client C 210 and Client D 212 are connected to the network 204. Communications with the server 202 are directed through a communications filter 214, which controls the number of clients that can access the server or utilize software on the server. The communications filter 214 monitors communications between the server 202 and the client computers 206 – 212 and is configured to identify a network address for each of the clients 206 – 212 that communicate with the server 202.

Although the communications filter 214 is shown as being separate from the server 202, the communications filter 214 may be located on the server 202 or on a peripheral unit connected to the server 202. Generally, the communications filter 214 may be located anywhere that it can monitor communications between

the server 202 and the clients 206 – 212 and where a client cannot bypass the communications filter 214 to communicate with the server 202.

The communications filter 214 may control access to the server 202 itself (i.e., to a server operating system), to a software application stored on the server 202, or to a hardware device (not shown) that is connected to the server 202. The present discussion will focus on a communications filter that is configured to limit the number of unique clients that can access and utilize a server operating system. Also, for purposes of the present discussion, reference made to accessing the server will mean accessing an operating system on the server 202.

Fig. 3 is a block diagram of a server-client network system 300 implemented in accordance with the described embodiments. The system 300 includes a server 302 that communicates with a network 306. Client A 308, client B 310, client C 312 and client D 314 communicate with the server 302 via the network 304. It is noted that, although only four clients are shown that communicate with the server 302, virtually any number of clients can communicate with the server 302. The number of clients accessing the server 302 is limited in the present invention only by a communications filter, which will be discussed in greater detail below.

Client A 308 includes a network interface card 318 (NIC) that is used to connect client A 308 to the network 306. Client A 308 also has a network address 320 that uniquely identifies client A 308 on the network 306. Client B 310 includes a network interface card 322 that connects client B 310 to the network 306. Client B 310 also has a network address 324 that uniquely identifies client B 310 on the network 306. Client C 312 includes a network interface card 326 to connect client C 312 to the network 306. Client C 312 also has a network address

328 that uniquely identifies client C 312 on the network 306. Client D 314 includes a network interface card 330 and has a network address 332 that uniquely identifies client D 314 on the network 306.

The server 302 includes a processor 334, a network interface card 336 and memory 338. The processor 334 is a central processing unit that is configured to execute processor-executable instructions. The network interface card 336 provides an interface between the server 302 and the network 306. The server 302 communicates with the network 306 using standard TCP/IP (Transmission Control Protocol/Internet Protocol) communications protocol.

It is noted that, although the server 302 is shown communicating via the network 306 using TCP/IP, the server 302 may also communicate over any network - such as a wide area network (WAN) - utilizing any appropriate protocol. The server 302 may also be configured to use a method other than a network interface card to interface with the network 306. For example, the server 302 may utilize a modem (not shown) to communicate with client computers in a wide area network.

The memory 338 includes an operating system 340, which has a global system registry 342. A client limit 344 is stored in the registry 342 and is, therefore, available for retrieval by various software applications. The client limit 344, as will be discussed in greater detail below, is a value that specifies a maximum number of unique clients that can access the server 302. For purposes of discussion, accessing the server 302 means accessing and utilizing the operating system 340, although the described implementations could be used to limit access to the server 302, a software application resident on the server 302, or some piece electronic hardware unit. The client limit 344 is the number of users allowed by,

for example, a concurrent users license. The client limit 344 is preferably configurable, so that a software designer or vendor can customize the client limit for each particular server. The configurable client limit 344 can be set to different limits on different server products.

The memory 338 also includes a protocol stack 346 that is used to process communication protocol commands received from one or more clients 308 – 314. The protocol stack 346 includes related communication protocol layers stacked on top of each other (data products, sessions, etc.)

A decryption module 348 resides in the memory 338 and is used when the client limit 344 is stored in an encrypted form. The encrypted client limit 344 is decrypted by the decryption module 348 and is stored in the memory as client limit 350. The decrypted client limit 350 is used in subsequent processing. In an alternative implementation, the decryption module 348 decrypts the encrypted client limit 344 whenever the client limit 344 is used in processing. However, it is preferable to utilize a decrypted client limit 350 in processing, as this economizes computational overhead.

The memory 338 also includes a communications filter 352 that is configured to limit the number of clients that can access the server 302 to the value indicated by the client limit 350. The communications filter 352 consists of a communications filter driver 354 and a communications filter controller 356. The communications filter driver 354, as will be discussed below, is configured to signal an event when the client limit 350 is exceeded. The communications filter controller 356 initializes the communications filter driver 354 and retrieves the client limit 350 (or the client limit 344 in the registry 342). The communications filter controller 356 is additionally configured to send a signal to all clients that

have previously accessed the operating system 340 and monitor for a response from each client. The purpose of this function will be discussed in detail, below.

It is noted that the preferred implementation, as described below, will identify the tasks that are allocated to and performed by the communications filter controller 356. However, it is noted that the communications filter controller 356 may also be configured to perform other tasks described in the following discussion, although such other tasks may not be specifically recited as being performed by the communications filter controller 356.

The communications filter driver 354, when initialized with the client limit 350, creates a client table 358 having a number of entries equal to the number of unique users allowed to access the operating system 340 under the applicable users license. The communications filter driver 354 then registers with the communications stack 346 to receive notification on every incoming packet on the local network interface card 336 (the local NIC is the network interface card through which clients on the network 306 connect to the server 302). The remaining functions of the communications filter driver 354 will be discussed in greater detail, below, with reference to Fig. 4.

The client table 358 has one entry available for each unique user that accesses the operating system 340. For example, if the client limit 350 is twenty-five (25), then the client table 358 will consist of twenty-five (25) entries, or rows. As shown in Fig. 3, the client table 358 comprises three (3) rows: row 360, row 362 and row 364. For purposes of this example, it is assumed that the client limit is three (3), although the client limit 350 could be virtually any number. In the present example, row 360 contains a network address associated with and uniquely identifying client A 308. The network address associated with client A 308 is

64.236.8.10. Row 362 contains a network address associated with client B 310, and row 364 contains a network address associated with client C 312. The network address for client B 310 is 64.220.16.27, and the network address assigned to client C 312 is 32.99.128.16.

It is noted that the network addresses stored in row 360, row 362 and row 364 will not actually be stored in the rows until client A, client B and client C are granted access to the operating system 340. However, for convenience purposes, the network addresses are shown stored in the rows 360 – 364. The process by which the network addresses are stored is discussed in detail, below, with reference to Fig. 4.

Fig. 4 is a flow diagram that depicts a method in accordance with the present invention. At step 400, the communications filter driver 354 monitors incoming communications packets that are sent across the local network interface card 336. If a communications packet is a packet is a session initiation packet- such as a TCP SYN (TCP synchronization) packet - ("Yes" branch, step 402), then the network address is identified from the communications packet at step 404. If the communications packet is not a session initiation packet - such as a TCP SYN packet - ("No" branch, step 404), then it is assumed that the client has previously accessed the server 302, and the packet is processed at step 416.

It is noted that the communications filter driver 354 could examine all communications packets received from a client. However, for performance reasons, in the preferred implementation only the session initiation packets are inspected to determine the network address of a sending client.

Once, the network address has been determined from the packet, the communications filter driver 354 determines if the network address is listed in the

client table 358. For discussion purposes, assume that at this point, the client table 358 contains no entries and that client A 308 has sent a TCP SYN packet to the server 302. At step 406, the communications filter driver 354 searches the client table 358 for the network address 320 of client A 308 (64.236.8.100). If the network address for client A 308 is found in the client table 358 ("Yes" branch, step 406), then client A 358 is allowed to access the server 302 and the packet is processed at step 416. But in this example, the client table 358 is empty so no match is found for the network address 320 of client A 308 ("No" branch, step 406).

At step 408, the communications filter driver 354 determines if the maximum number of clients that are allowed to access the operating system 340 has been reached. The client limit 350 is retrieved from memory 338. If the number of entries in the client table 358 is less than the client limit 350 ("No" branch, step 408), then the network address 320 for client A 308 is stored in the client table 358 at row 360 (step 410) and the packet is processed at step 416.

At this point, client A 308 has access to the server operating system 340. Continuing with the example, now assume that client B 310 attempts to access the operating system 344 of the server 302. At step 402, the communications filter driver 354 detects a data packet that includes the network address 324 of client B 310. The network address 324 is identified at step 404 and, at step 406, the communications filter driver 354 compares the network address 324 for client B 310 (64.220.16.27) to the entries in the client table 358 in an attempt to find a match. Since client B 310 has not previously accessed the operating system 344, the communications filter determines if the client limit 350 has been reached by comparing the number of entries in the client table 358 to the client limit 350. In

this example, the client limit 350 is three (3) and the number of entries in the client table 358 is one (client A). Therefore, the limit has not been reached ("No" branch, step 408) and the network address 324 for client B 310 is stored in the client table 358 at step 410. The communications packets received from client B 310 are processed at step 416.

At this point, client A 308 and client B 310 have access to the server operating system 340, and the network addresses 320, 324 for client A 308 and client B 310 are stored in the client table 358. Continuing with the example, client C 312 attempts to access the operating system 344 of the server 302. At step 402, the communications filter driver 354 detects a data packet that includes the network address 328 of client C 312. The network address 328 is identified at step 404 and, at step 406, the communications filter driver 354 compares the network address 328 for client C 312 (126359.44.33) with the entries in the client table 358 to find a match. Since client C 312 has not previously accessed the operating system 344, the communications filter determines if the client limit 350 has been reached by comparing the number of entries in the client table 358 to the client limit 350. The client limit 350 is three (3) and the number of entries in the client table 358 is now two (clients A and B). Therefore, the client limit has not been reached ("No" branch, step 408) and the network address 328 for client C 312 is stored in the client table 358 at step 410. The communications packets received from client C 312 are processed at step 416.

Assume now that client D 314 attempts to communicate with the operating system 344 of the server 302. At step 402, the communications filter driver 354 detects a TCP SYN packet sent from client D 314 and the network address 332 for client D 314 is identified. The network address 332 for client D 314 is not in the

client table 358 ("No" branch, step 406), so the client limit is checked at step 408. Now, the number of entries (or network addresses) in the client table 358 is three, which is equal to the client limit 350. Therefore, client D 314 is not immediately allowed access to the operating system 344.

As previously discussed, there are some systems that are configured to reassign a network address for a client when the network address for the client has expired for one of various reasons. For example, a system might reclaim a network address from a client if the client has not been active for a certain period of time. If the client becomes active again at a later time, the system assigns a new network address to the client. This new network address may be identical to the original network address, but it can be a different network address. Therefore, the present invention must accommodate this type of scheme.

At step 412, the communications filter driver 354 attempts to determine if client D 316 has previously accessed the operating system 344 using a different network address. If client D 316 has previously accessed the server 302 using a different network address ("Yes" branch, step 412), then the new network address used by client D 314 is substituted for the old network address used by client D 314 (step 414). But if client D 314 has not previously accessed the server 302, then at step 418, a limit exceeded event is initiated wherein the communications filter driver 354 notifies the communications filter controller 356 that the limit has been exceeded. The communications filter controller 356 sends a "limit exceeded" signal to the operating system 340 (and which is ultimately sent to the client) and the packet is dropped. In other words, client D 316 is not allowed to communicate with the server 302.

One way in which the communications filter 352 determines if the new client (client D 316) has previously accessed the server 302 is by the communications filter controller 356 sending a signal to each network address listed in the client table 358. If a client at one network address fails to acknowledge the signal, then the communications filter controller 356 assumes that the network address is no longer valid and, therefore, another client may access the server 302 within the client limit 350. When a response is not received from a network address, the communications filter controller 356 removes the non-responsive network address from the client table 358 and inserts the new network address in its place (step 414). Therefore, the client limit 350 is still not exceeded.

It is noted that a client may not respond because it has been temporarily shut down. If so, then it is theoretically possible to exceed the client limit. However, it would be very inconvenient for users to do this simply to exceed a concurrent users license and it is only a small probability that such a technique would, in practice, actually be used to exceed the limit. That notwithstanding, the described implementations still provide a reliable, practical technique for keeping the number of clients that access the server within the client limit.

Conclusion

The systems and methods described herein provide a practical way to limit the number of clients that can access a server in a server-client network system. The server keeps track of each network address from which it is accessed and denies access to any client that attempts to access the server after the maximum number of clients have accessed the server.

Systems that allow network address to expire and that reassign network addresses to existing clients can be implemented as well. When a client attempts to access the server when the client limit has been reached, the server signals each network address that identifies a client having access to the server. If a network address does not acknowledge the signal, then the server assumes that the network address is no longer in use, and the server allows another client to access the server.

Although details of specific implementations and embodiments are described above, such details are intended to satisfy statutory disclosure obligations rather than to limit the scope of the following claims. Thus, the invention as defined by the claims is not limited to the specific features described above. Rather, the invention is claimed in any of its forms or modifications that fall within the proper scope of the appended claims, appropriately interpreted in accordance with the doctrine of equivalents.

CLAIMS

1. An Internet protocol (IP) filter, comprising processor-executable instructions that, when executed on a processor, perform the following steps:

monitoring Internet protocol data packets transmitted from one or more clients to a server;

obtaining a network address from an IP data packet transmitted by a client; and

processing IP data packets from the client if a Network address that is uniquely associated with the client is stored in a client table.

2. The Internet protocol filter as recited in claim 1, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

if the Network address is not stored in the client table, retrieving a client limit value from a client limit field, the client limit value indicating a maximum number of unique clients for which IP data packets can be processed;

processing IP data packets from the client if the number of Network addresses in the client table is less than the client limit value; and

storing the Network address in the client table.

3. The Internet protocol filter as recited in claim 1, wherein the client is a first client and the Network address is a first Network address, the Internet protocol filter further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

if the first Network address is not stored in the client table, retrieving a client limit value from a client limit field, the client limit value indicating a maximum number of unique clients for which IP data packets can be processed;

if the number of Network addresses in the client table is greater than or equal to the client limit value, determining if the first client is represented in the client table by a second Network address that is different from the first Network address; and

processing IP data packets from the first client if the second Network address is found in the client table.

4. The Internet protocol filter as recited in claim 3, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

removing the second Network address from the client table; and

inserting the first Network address into the client table.

5. The Internet protocol filter as recited in claim 1, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

if the first Network address is not stored in the client table, retrieving a client limit value from a client limit field, the client limit value indicating a maximum number of unique clients for which IP data packets can be processed;

if the number of Network addresses in the client table is greater than or equal to the client limit value, transmitting a signal to each Network address listed in the client table; and

if a client at a second Network addresses does not respond to the signal, removing the second Network address from the client table, inserting the first Network address into the client table and processing IP data packets from the first client.

6. The Internet protocol filter as recited in claim 5, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

removing the second Network address from the client table; and

inserting the first Network address into the client table.

7. A method, comprising:

detecting when a current client attempts to establish a connection with a server;

determining a unique client identifier that is associated with the current client;

determining if a total number of previous clients having access to the server is less than a client limit;

determining if the current client has previously been allowed to access the server;

providing access to the server if the total number of previous clients having access to the server is less than a client limit;

providing access to the server if the total number of previous clients is greater than or equal to the client limit and if the current client has previously been allowed to access the server; and

storing the unique client identifier associated with the current client in memory if access is provided to the current client.

8. The method as recited in claim 7, wherein the determining if a total number of previous clients having access to the server is less than a client limit further comprises:

determining how many unique identifiers are stored in memory; and

comparing the number of unique identifiers in memory with the client limit.

9. The method as recited in claim 7, wherein the determining if the current client has previously been allowed to access the server is only performed if the total number of previous clients having access to the server is greater than or equal to the client limit.

10. The method as recited in claim 7, wherein the determining if the current client has previously been allowed to access the server further comprises:

comparing the unique identifier of the current client with the unique identifiers of each previous client that has been allowed to access the server;

determining that the current client has previously been allowed to access the server if the current client identifier matches a previous client identifier.

11. The method as recited in claim 7, wherein the determining if the current client has previously been allowed to access the server further comprises:

transmitting a signal to each previous client that has been allowed to access the server; and

determining that the current client has previously been allowed to access the server if at least one of the previous clients fails to acknowledge the signal.

12. The method as recited in claim 7, further comprising:

pre-configuring the client limit; and

storing the client limit in memory.

13. The method as recited in claim 12, wherein the client limit has a pre-defined maximum to which it may be configured.

14. The method as recited in claim 7, further comprising:

pre-configuring the client limit;

encrypting the client limit; and

storing the encrypted client limit in memory.

15. The method as recited in claim 7, further comprising:

retrieving an encrypted client limit; and

decrypting the encrypted client limit to derive the client limit.

16. The method as recited in claim 7, wherein the determining the unique client identifier that is associated with the current client further comprises identifying an Internet protocol address from a data packet transmitted by the current client.

17. The method as recited in claim 7, further comprising storing the unique client identifiers in a client table in memory.

18. The method as recited in claim 7, wherein the client identifier is a network address.

19. A server that provides access to a limited number of clients, comprising:

memory;

a network interface configured to handle communications between the server and a plurality of clients;

an operating system stored in the memory;

a client limit stored in the memory, the client limit denoting a number of unique clients that are allowed to access the server;

an IP stack in the memory that is used to process data packets transmitted from clients;

a client table in the memory for storing a unique Network address for each client that accesses the server; and

a communications filter configured to allow access to a first client if the total number of clients that have accessed the server is less than the client limit, or if the total number of clients that have accessed the server is greater than or equal to the client limit and the first client has previously accessed the server.

20. The server as recited in claim 19, wherein the Communications filter is further configured to search the client table for a first Network address associated with the first client and determine that the first client has previously accessed the server if the first Network address is found in the client table.

21. The server as recited in claim 19, wherein the Communications filter is further configured to search the client table for a second Network address associated with the first client and determine that the first client has previously accessed the server if the second Network address is found in the client table.

22. The server as recited in claim 21, wherein the Communications filter is further configured to determine the second Network address by signaling each Network address listed in the client table and determine that the second Network address is a network address listed in the table that does not acknowledge the signal.

23. The server as recited in claim 19, wherein the client limit is configurable.

24. The server as recited in claim 19, wherein the Communications filter is further configured to signal that the client limit has been exceeded and to deny server access to the first client if the total number of clients that have accessed the server is greater than or equal to the client limit, and the first client has not previously accessed the server.

25. The server as recited in claim 19, wherein the client limit is encrypted, the server further comprising a decryption module configured to decrypt the encrypted client limit.

26. A method for providing server access to a limited number of clients, the method comprising:

monitoring TCP/IP packets sent from a plurality of clients to a server;

obtaining a unique Network address for each client from one or more packets transmitted by the client;

storing the Network address of each client that accesses the server;

determining if a client limit has been reached; and

providing access to a first client if the client limit has been reached, or if the first client has previously accessed the server.

27. The method as recited in claim 26, wherein the determining if the client limit has been reached further comprises:

determining how many unique clients have accessed the server;

comparing the number of unique client with the client limit; and

determining that the client limit has been reached if the number of unique clients is greater than or equal to the client limit.

28. The method as recited in claim 26, wherein the determining if the first client has previously accessed the server further comprises:

comparing a first Network address that uniquely identifies the first client with a table of stored Network addresses; and

determining that the first client has previously accessed the server if the first Network address matches a stored Network address.

29. The method as recited in claim 26, wherein the determining if the first client has previously accessed the server further comprises:

sending a signal to each of multiple Network addresses of clients that have accessed the server; and

if there is no response to one of the signals, determining that the first client has previously accessed the server using the Network address of the client from which there was no response detected.

30. The method as recited in claim 26, further comprising:

retrieving an encrypted client limit; and

decrypting the encrypted client limit to derive the client limit.

31. An operating system stored on a computer-readable medium, the operating system comprising:

an IP stack for processing Internet protocol data packets received from multiple clients;

a client limit field containing a client limit value that denotes a maximum number of clients that may access the IP stack;

a client table containing a unique Network address for each client that has accessed the operating system; and

a communications filter configured to determine a first Network address of a first client attempting to access the operating system, search the client table for the first Network address, and allow the first client to access the operating system if the first Network address is found in the client table.

32. The operating system as recited in claim 31, wherein the Communications filter is further configured to:

allow the first client to access the operating system if the number of Network addresses in the client table is less than the client limit value; and

store the first Network address in the client table if the first client is allowed to access the operating system.

33. The operating system as recited in claim 31, wherein the Communications filter is further configured to allow the first client to access the operating system if the number of Network addresses in the client table is greater than or equal to the client limit value and the first client has previously accessed the operating system using a second Network address that is stored in the client table.

34. The operating system as recited in claim 33, wherein the Communications filter is further configured to:

transmit a signal to each Network address listed in the client table;

monitor for an acknowledgement to each signal; and

if an acknowledgement is not received from a network address in the client table, determining that the non-acknowledging Network address is the second Network address used by the first client.

35. The operating system as recited in claim 34, wherein the Communications filter is further configured to replace the second Network address in the client table with the first Network address.

36. The operating system as recited in claim 31, wherein the client limit value is encrypted, and the operating system further comprises a decryption module that is configured to decrypt the client limit value.

37. A computer-readable medium comprising computer-executable instructions that, when executed on a computer, perform the following steps:

determining a first Internet Protocol (IP) address transmitted from a first client to a server;

searching a client table for the first Network address; and

allowing the first client to access the server if the first Network address is found in the client table.

38. The computer-readable medium as recited in claim 37, further comprising computer-executable instructions that, when executed on a computer, perform the following steps:

determining if a client limit has been reached, the client limit indicating a total number of clients that can access the server;

allowing the first client to access the server if the client limit has not been reached; and

inserting the first Network address into the client table.

39. The computer-readable medium as recited in claim 37, further comprising computer-executable instructions that, when executed on a computer, perform the following steps:

transmitting a signal to each Network address listed in the client table; and

if there is no response from one of the Network addresses signaled, allowing the first client to access the server, removing the non-responsive Network address from the client table, and inserting the first Network address into the client table.

40. A computer system, comprising:

a processor;

a network interface card to handle communications with multiple clients;

memory;

a global system registry;

a client table having one entry for each client allowed to access the system, each entry including a unique Internet protocol (IP) address for each client; and

a communications filter configured to:

retrieve a client limit from the global system registry;

determine a first Network address that is associated with a first client attempting to access the system;

allow the first client to access the system if the first Network address is stored in the client table or if the number of client table entries is less than the client limit; and

store the first Network address in the client table if the first client is allowed to access the system.

41. The computer system as recited in claim 40, wherein the Communications filter is further configured to allow the first client to access the system if the number of entries in the client table is greater than or equal to the client limit and if the first client has previously accessed the system.

42. The computer system as recited in claim 41, wherein the Communications filter is further configured to determine if the first client has previously accessed the system if the first Network address is stored in the client table.

43. The computer system as recited in claim 41, wherein the Communications filter is further configured to determine if the first client has previously accessed the system by transmitting a signal to each Network address listed in the client table, monitoring responses to the signals to determine if a client at a second Network address is no longer using the Network address, substituting the first Network address in the table for the second Network address and allowing the first client to access the system if the client at the second Network address does not respond to the signal.

44. The computer system as recited in claim 40, wherein the client limit is encrypted, the computer system further comprising a decryption module configured to decrypt the encrypted client limit.

45. A communications protocol filter, comprising processor-executable instructions that, when executed on a processor, perform the following steps:

monitoring communications protocol data packets transmitted from one or more clients to a server;

obtaining a network address from a communications protocol data packet transmitted by a client; and

processing communications protocol data packets from the client if a Network address that is uniquely associated with the client is stored in a client table.

46. The communications protocol filter as recited in claim 45, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

if the Network address is not stored in the client table, retrieving a client limit value from a client limit field, the client limit value indicating a maximum number of unique clients for which communications data packets can be processed;

processing communications protocol data packets from the client if the number of Network addresses in the client table is less than the client limit value; and

storing the Network address in the client table.

47. The communications protocol filter as recited in claim 45, wherein the client is a first client and the Network address is a first Network address, the communications protocol filter further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

if the first Network address is not stored in the client table, retrieving a client limit value from a client limit field, the client limit value indicating a maximum number of unique clients for which communications protocol data packets can be processed;

if the number of Network addresses in the client table is greater than or equal to the client limit value, determining if the first client is represented in the client table by a second Network address that is different from the first Network address; and

processing communications protocol data packets from the first client if the second Network address is found in the client table.

48. The communications protocol filter as recited in claim 47, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

removing the second Network address from the client table; and

inserting the first Network address into the client table.

49. The communications protocol filter as recited in claim 45, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

if the first Network address is not stored in the client table, retrieving a client limit value from a client limit field, the client limit value indicating a maximum number of unique clients for which communications protocol data packets can be processed;

if the number of Network addresses in the client table is greater than or equal to the client limit value, transmitting a signal to each Network address listed in the client table; and

if a client at a second Network addresses does not respond to the signal, removing the second Network address from the client table, inserting the first Network address into the client table and processing communications protocol data packets from the first client.

50. The communications protocol filter as recited in claim 49, further comprising processor-executable instructions that, when executed on a processor, perform the following steps:

removing the second Network address from the client table; and

inserting the first Network address into the client table.

51. The communications protocol filter as recited in claim 45, wherein the communications protocol is an Internet protocol and the communications protocol data packets are Internet protocol data packets.

ABSTRACT

Systems and methods for controlling the number of clients that can access a server in a server-client network are described. A communications filter driver resides on the server and monitors network communications to determine a unique network address of each client that attempts to access the server. A table of clients that have accessed the server is stored at the server. When a client attempts to access the server, the server allows the client to access the server if the network address of the client is listed in the table. If the network address of the client is not listed in the table, the server determines if a client limit has been met or exceeded. If the limit has not been met, the client is allowed to access the server and the Internet address of the client is added to the table.

If the limit has been reached, the server determines if the client is a client that has previously accessed the server under a different network address. This is accomplished by sending a signal to each network address listed in the table and requesting a response. If a response is not received from a client, then the network address for that client is removed from the table and the client attempting to access the server is allowed to access the server. The network address for this client is then added to the table. If a response is received from all network addresses, a signal indicating that a client limit has been exceeded is sent to the client attempting to access the server and access to the server is denied to that client.

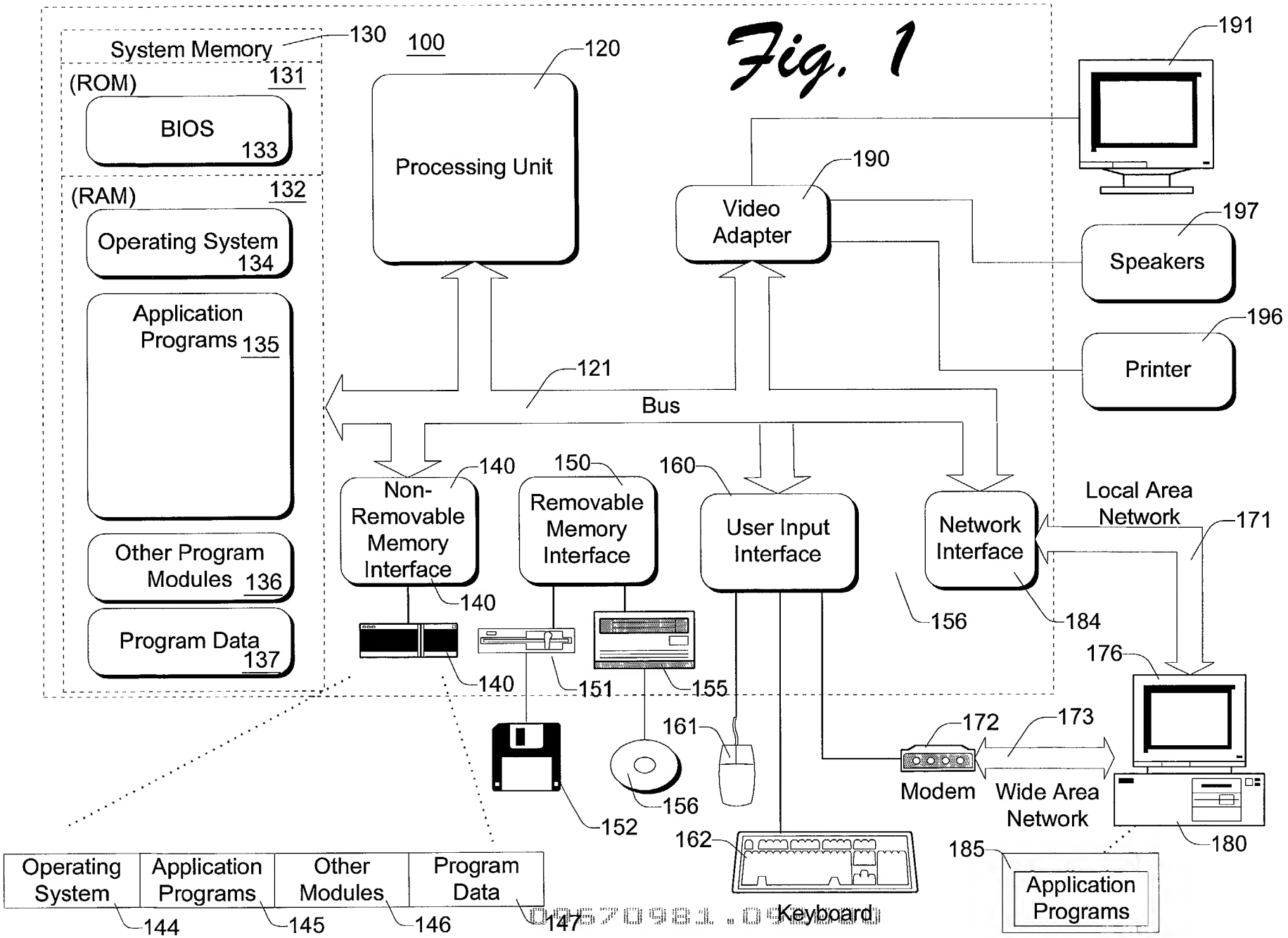
Fig. 1
System Memory
130
100
(ROM)
131
BIOS
133
(RAM)
132
Operating System
134
Application Programs
135
Other Program Modules
136
Program Data
137
Processing Unit
120
Video Adapter
190
191
Speakers
197
Printer
196
121
Bus
Non-Removable Memory Interface
140
Removable Memory Interface
150
User Input Interface
160
Network Interface
Local Area Network
171
156
184
140
140
151
155
161
172
173
176
Modem
Wide Area Network
180
152
156
162
185
Application Programs
Keyboard
Operating System
144
Application Programs
145
Other Modules
146
Program Data
147

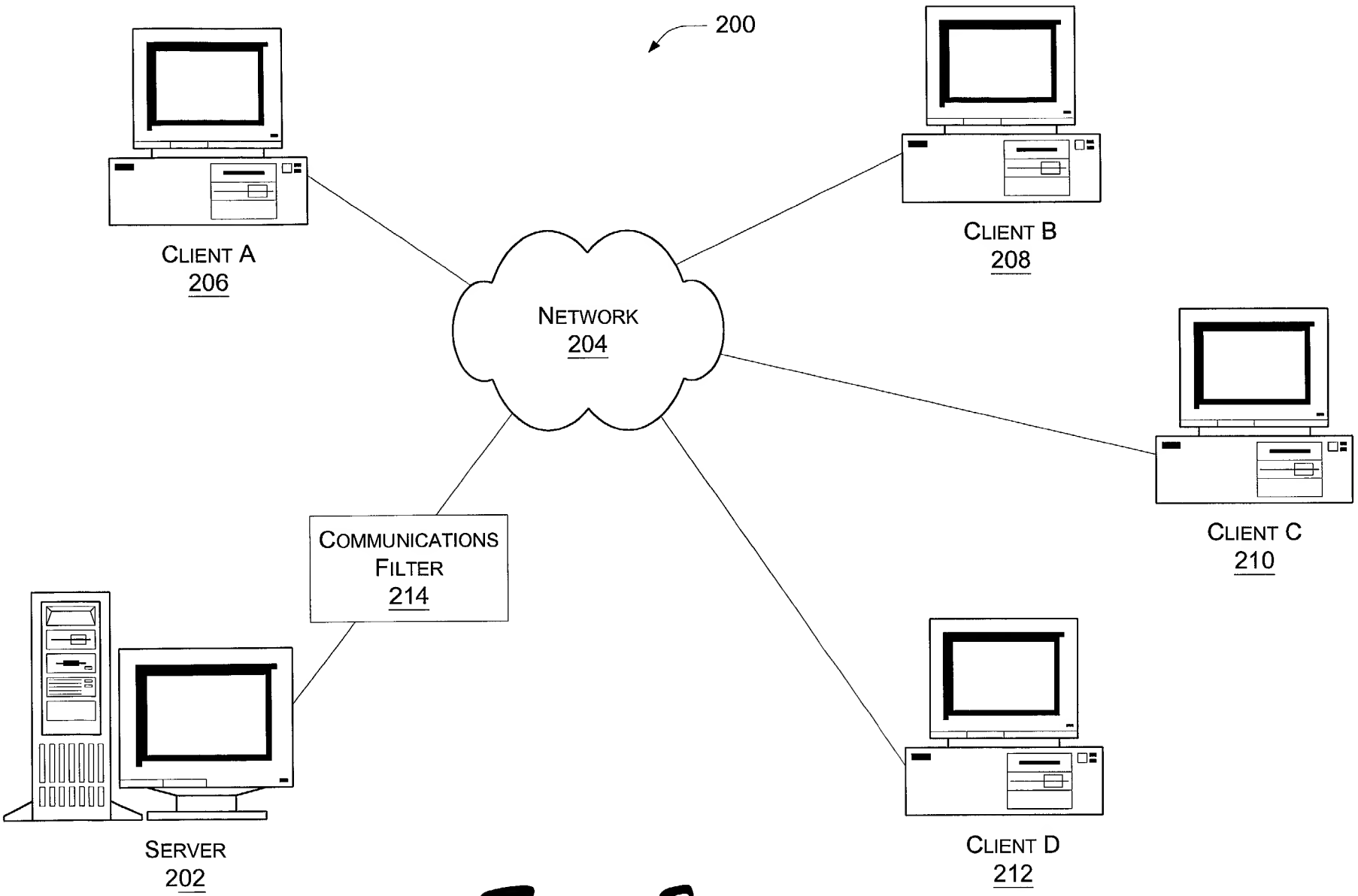

Fig. 2

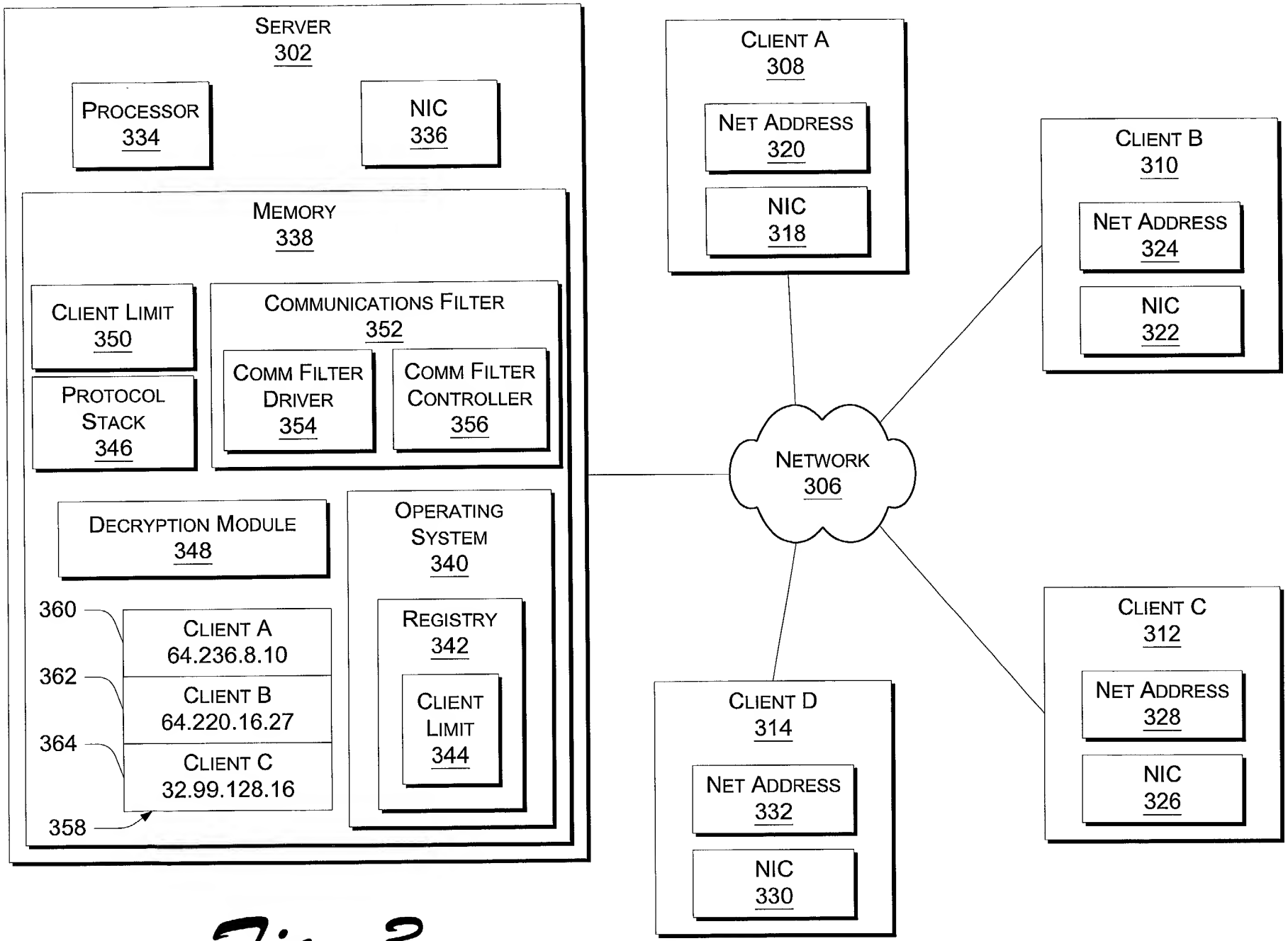

Fig. 3

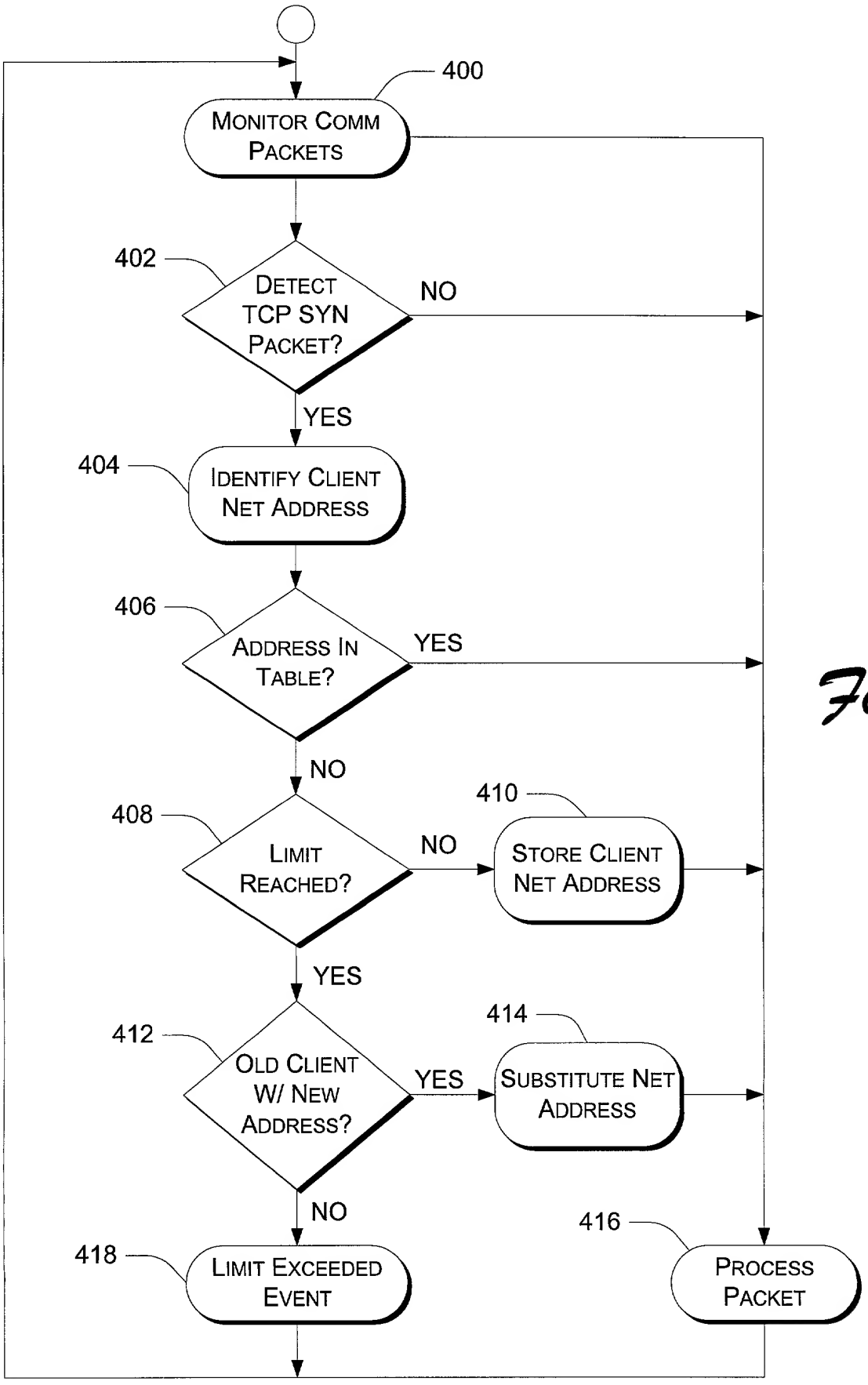

Fig. 4